

The RoboCup F-180 League

Dedicated System Design for Performance Analyses of Distributed Embedded Systems

K. Köker *, R. German **

*/** University of Erlangen-Nuremberg/Department of Computerscience 7,
Computer Networks and Communication Systems, Erlangen, Germany
{koeker, german}@informatik.uni-erlangen.de

Abstract—We present our work in progress for a new system architecture of the RoboCup F-180 small size league to analyse the performance of distributed embedded systems. Our system design allows shifting the autonomy from the centralized system to each of the soccer robots for locally execution of the strategy algorithm and autonomous action commands. The new system design dedicates the master host system as a global vision system for object recognition and broadcasting coordinates via WLAN. Since the acting soccer robots are not remote controlled anymore like in previous robot soccer systems in the F-180 league, we designed a new architecture with an embedded system on board with respect to the limitation of the robots dimension. The robots` embedded system has capabilities to process communication via WLAN and execute strategy applications in a fast way. Furthermore local sensors are implemented to assist the robots to recognize environment characteristics. Our architecture is aimed to conduct measurements based on hard- and software solutions and allows easily monitoring and evaluating the performance of distributed embedded systems in dynamic scenarios. The results should allow building simulation models for different system parameters to estimate system response in varying scenarios.

Key words: *embedded distributed systems, embedded Linux, performance analysis*

I. INTRODUCTION

RoboCup [1] is the soccer world championship for autonomous soccer robots, multi-agent systems and artificial intelligence. The Small-Size league, also called the F-180 league, is one of currently five RoboCup league divisions. The F-180 league focuses on the problem of intelligent multi-agent cooperation and control in a highly dynamic environment with a hybrid centralized/distributed system.

A game of the Small-Size soccer robots take place between two teams, whereat each team consists of five robots. Each robot must conform to the F-180 rules, whereby the robots must fit within a 180mm diameter circle and are not allowed to be higher than 15cm unless they use an on-board vision. The robots play soccer on a green carpeted field that is 4.9m long by 3.4m wide with an orange golf ball of 43mm diameter and 46g of weight.

Soccer robots in this league are designed with local on-board vision sensors or with a global vision system.

By far the most common system variety uses a global vision system for the robots, to identify and track the robots as they move around the field. In this variety an overhead camera is attached to a camera bar located 4m above the playing surface, and connected again to an off-field host to perform object recognition. Typically the off-field computer also performs almost all of the processing which is required for the coordination and control of the robots. The communication between the off-field host and the robots is usually wireless and typically uses dedicated commercial FM transmitter/receiver units for action commands. Figure 1 illustrates a typically system design for the F-180 league with remotely controlled soccer robots by the off-field host. In the past, researches and developments aimed to improve the mechanic, electronic, pattern recognition and to implement new strategy programs for the success. In the F-180 league e.g. the processing time for the vision system and the strategy algorithms also may decrease due to increasing the computing power of the host system.

The commonly used centralized system design may provide a successfully game, but doesn't allow a detailed system analysis, when the robots should act as distributed autonomous units. Thus, the common design of a F-180 league system isn't appropriate to analyse the response time for an autonomous soccer robot in detail e.g. avoiding collision with an opponent or the boundary.

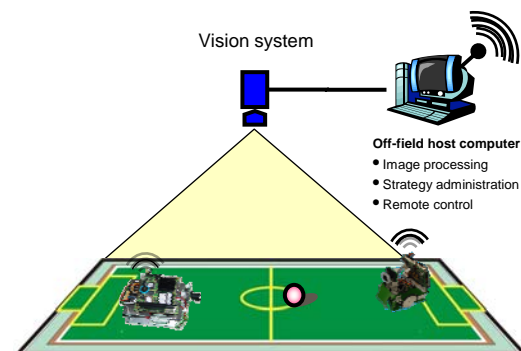


Fig. 1. RoboCup F-180 system using remote control

Our goal is to analyse the performance of distributed networked embedded systems in high dynamic environments. For this purpose we redesigned the system architecture of the F-180 league to easy conduct monitoring, measuring and modelling entirely. The modelling of the entire system is aimed to detect bottlenecks, response delays and to evaluate the performance and behaviour of the architecture for the purpose in dynamic environment as described below. The design of the system and the workings are still in progress but allow by now to track the ball and perform single measurements for the system response for the robot.

II. SYSTEM ARCHITECTURE

Our system consists of 2 main instances: an off-field host computer for the vision system like in common F-180 league systems, and the autonomous robots with embedded systems on board.

A. Vision System

For the global vision system, we connected a TMC-6 DSP CCD camera from Pulnix [2] with a resolution of 752 (horizontal) x 582 (vertical) pixels to the off-field computer and attached it to a bar about 3 meters above the play field. The camera is connected via a frame grabber with BT878 chip onboard to the host computer with the SuSe-Linux 10.1 [3] operating system. The implementation used here is capable to work at a frame rate of 70 Hz and is based on the implementation from [4], whereby the section for the strategy is changed.

Objects on the play field are recognized and distinguished by the vision system using coloured squares mounted on the top of the robots. The markings vary with respect to the team and the team members. Figure 2 illustrates the marking for a team member of the yellow team, which is indicated by a wide yellow rectangle on the lower half of the colour square which also marks the rear of the robot. The implementation for the object recognition is aimed to gain information about the coordinates of each marked object with respect to the origin and also the angle with respect to the axis line of the side boundary.

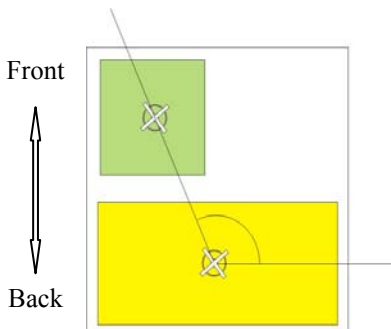


Fig. 2. Robots color marking with the reference line to obtain the angle of the robot

After all coordinates and available angles of the objects are determined by the vision system, the off-field computer composes a state matrix with the values for the transmission for the i -th point of time and the j -th object (Equation 1).

$$M_{i,j} = (x_{i,j} \quad y_{i,j} \quad \alpha_{i,j}) \quad (1)$$

The transmission of the matrix is performed every 10ms via UDP broadcast over WLAN (IEEE 802.11g). Due to the CSMA/CA access mode for the medium, it's not appropriate that multiple clients share the same channel for other communication when the matrix has to be received right on time. Usually, each time before a client starts to send, he has to check the clearance of the medium which leads to a non deterministic time for collision free data transmission. Therefore the clients are just listening to the matrix broadcast of the host computer and remain quiet. After receiving the matrix, each robot has to locally execute the strategy application for the next movement. The process of composing, broadcasting and correct receiving of the matrix is a precondition for the robots to act autonomously.

B. Soccer Robots

The architecture of a soccer robot in the F-180 league usually consists of the mechanical and electronic part. The mechanical part covers the body and the actors. The electronic part consists of a microcontroller for the actors, a dedicated FM-module to receive action commands from the host, and the power supply with batteries. Usually this architecture disallows autonomous behaviour like path planning or collision avoidance because each action command is triggered from the host computer.

1) Robots Hardware

We modified the commonly used system by providing each player with an embedded PC/104 [5] system with capabilities for local execution of the strategy application. The object recognition is still handled by the host system which also broadcasts the matrices for the coordinates via WLAN to the clients.

The PC/104 system is a small size common embedded hardware in industrial projects with the dimension of almost 10x10cm. Although the PC/104 is very small in size, it's similar to an off-the-shelf desktop computer including onboard interfaces for the serial, parallel and USB port and also interfaces for a LAN, hard disk and floppy drive. For the robots we used the PC/104 module from Arbor [6] with an AMD Ultra Low Power Geode GX1-300-Mhz fan less CPU and an onboard compact flash socket.

Figure 3 displays our 2 layer architecture for the soccer robots whereby the upper layer is dedicated as the communication and computing instance. The first task of this instance is the correct reception of the matrix $M_{i,j}$ and extraction of the required data to provide the strategy application.

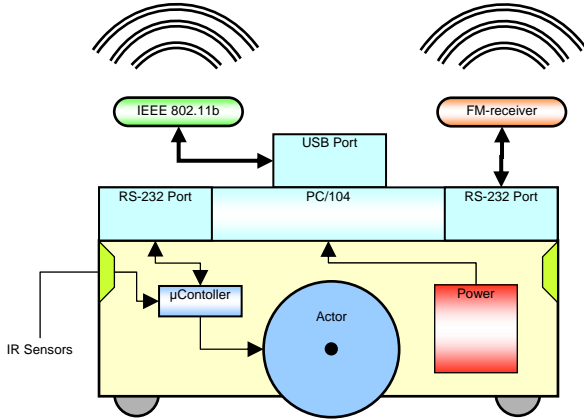


Fig. 3. Autonomous F-180 soccer robot

The second task is to execute the application to obtain the next move and to prepare the action commands to the actors. In the third and last step, the action commands are transmitted via the serial port to the actors, which are located on the lower layer.

The lower layer of the robots consists of an ATMEGA-Atmega32 [7] micro controller, infrared sensors, DC motors as actors and the power supply. The microcontroller converts analogue values of the sensors (infrared, system power) for sending them to the upper layer and drives the actors according to the received command from the serial port of the upper layer with the PC/104 module. The infra red sensors are intended for assistance of the strategy and drive application to avoid collisions with other robots and the boundaries. Therefore, the system response for the upper layer has to be fast to be assisted by the infra red sensors.

2) Operating System

We build an embedded Linux system for the PC/104 board with the Linux kernel version 2.6.9 [8]. To keep the system embedded, we used the popular BusyBox [9] as a tiny multi-call binary combining many common Linux/UNIX utilities. For the C library of the system we used uClibc [10] which is an optimized small C library for developing embedded systems common to the GNU C library.

Using these components for our system, allows keeping the system small and truly embedded. The fully functionally embedded Linux system is compressed and stored on a compact flash card and uses 3,400 Kbytes of space including the kernel. Due to the limited write-cycles of a compact flash card, the system is running in ramdisk mode by loading a compressed initial ramdisk on booting. Further details and an overview of the storage size for the fully functionally embedded Linux system can be found at the chairs website in [11].

3) Real Time Operation System

Using infra red sensors to avoid collisions with other robots or boundaries is reasonable to assist the robot performing fast moving. The assistance by the sensors depends on a fast system response for the interrupt caused by sensor data input. Real-time operating systems guarantee a maximum time for the systems response for interrupt and represent a quite interesting possibility to be used in soccer robots. Smart scheduling algorithms and other techniques are used to keep the response time as low as possible. We looked for a free and open source real time version of Linux to keep the project cost efficient and selected the real-time application interface RTAI. RTAI is an extension and modification of the common Linux kernel. Patching the kernel with RTAI enables the operating system to response in a fast and predictable way. Fast means that it has low latency, thus i.e. it responds to external, asynchronous events in a short time. Predictable means in that case that it is able to determine task's completion time with certainty. Further details for RTAI are available at [12].

III. EXAMPLE PERFORMANCE ANALYSES OF ROBOTS' OPERATING SYSTEM

The architecture design of the robot allows easily analyse the performance of the real time operating system. For a brief overview of the system performance, we set up a test bed and used a signal generator to generate rectangle signals at TTL level as system input. The input signal was connected to pin 10 of the PC/104s parallel port and caused an interrupt. Each time a high level is on the pin, an interrupt service routine (ISR) on the PC/104 is called up and the corresponding response is served at pin 2-9. The system response was monitored by the 4-channel, digital oscilloscope WaveSurfer 424 from LeCroy [4].

Figure 4 illustrates the hardware based measurement infrastructure for the first test run. Figure 5 displays a screenshot of the oscilloscope with a latency of $4.3 \mu\text{s}$ for the response time at 10 kHz signal input. The recording is truncated only for one signal from the frequency generator.

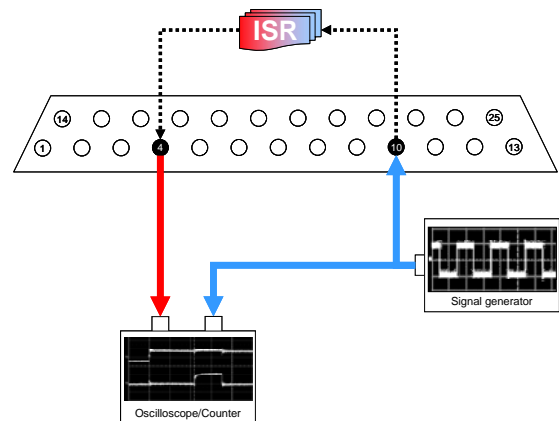


Fig. 4 Monitoring infrastructure via oscilloscope/counter

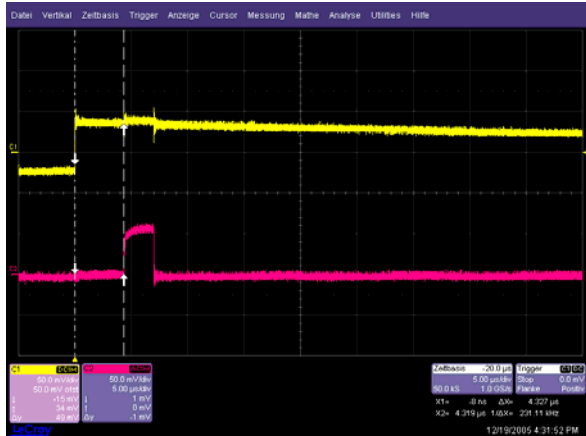


Fig. 5 Screenshot of the oscilloscope – latency for response 4.3 μ s

Our lab construction allows a screenshot of the system in various modes with different signal frequencies and system loads. Detailed analyses of the performance require to record time-related data for our construction for a long time period. For this purpose we used the high-precision frequency counter SIS3820 from [14] to record relevant data. According to the monitoring infrastructure in figure 4, we connected the start port of the frequency counter with the output of the signal generator to initiate the measuring. At the same time, the signal output is connected to pin 10 of the PC/104's parallel port to cause an interrupt that has to be handled by the ISR. The stop port of the counter is connected to one of the response pins of the parallel port. The counter uses an internal 50-MHz clock and therefore has a resolution of 20 ns. To measure the time between the start and stop signal, the counter is counting the ticks between the start and stop signal. We used our lab construction to record 3.6 million interrupts for the real-time mode (one-shot or periodic) and the system state (idle or load). The test durations lasts 6 minutes for each test mode at a signal input frequency of 10 kHz. To be able to compare the results, we increased the PC/104s system load by executing a ping flood via LAN from an external computer and recorded the response times for the interrupts. The statistical evaluation result, that the real-time operating system should preferably run in periodic mode scheduling for best performance for the response in system load state (Figure 6).

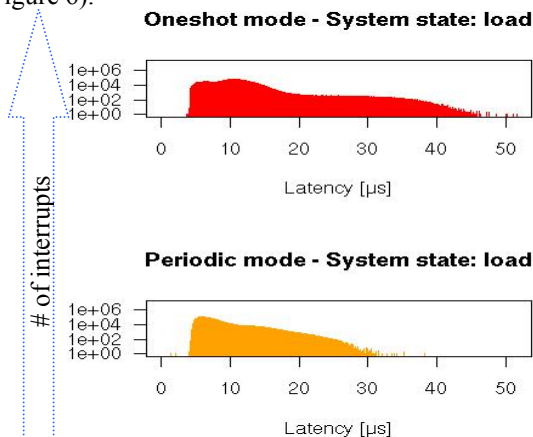


Fig. 6 Interrupt latency for the load state

IV. CONCLUSION AND FURTHER WORK

We presented a system architecture for the RoboCup F-180 league to analyse the performance of distributed embedded systems. For this purpose, we provide each robot with an industrial PC/104 embedded system with embedded Linux on board. The robots are equipped with distance sensors to assist the strategy application to compute the best moving path. The sensors transmit values via the serial port to the embedded system which has to respond as fast as possible and in deterministic time. For this purpose we used the real-time extension RTAI for Linux. For a first performance evaluation of the embedded operating system, we monitored and analysed the system response for signal interrupts at the parallel port. The statistical analysis of our test conclude the periodic scheduling mode as the preferably mode for best results for the system response. In the next step, we will analyse the system performance focusing the serial port for the communication with the lower robot layer and the USB port for the matrix input for the strategy application. In the future the vision system will be analysed to gain information about the system performance in various frame rates of the camera and various system load for the visioning application.

V. REFERENCES

- [1] RoboCup, <http://www.robocup.org>, visited 15/01/2007
- [2] Pulnix, <http://www.pulnix.com>, visited 12/12/2006
- [3] Novell SuSe, <http://www.novell.com/linux/suse/select.html>, visited 20/01/2007
- [4] University of Plymouth UK, <http://www.tech.plym.ac.uk/robofoot/index.html>, visited 28/01/2007
- [5] PC/104 EMBEDDED CONSORTIUM, Specification v2.5, <http://www.pc104.org>, visited on 12/06/2004
- [6] ARBOR INC PC/104-module Em104-n513/VL, <http://www.arbor.com.tw>, visited 12/12/2005
- [7] ATMEL Corporation, <http://www.atmel.com>, visited 20/10/2006
- [8] Linux kernel archives, <http://www.kernel.org>, visited 05/06/2006
- [9] Busybox multcall binary, <http://www.busybox.net>, visited 10/06/2006
- [10] uClibc embedded C library, <http://uclibc.org>, visited 11/06/2006
- [11] RoboCup7 performance analyses of distributed embedded systems, <http://www7.informatik.uni-erlangen.de/~koeker/research/>, visited 28/02/2007
- [12] RTAI - RealTime Application Interface for Linux, <https://www.rtai.org>, visited 25/07/2006
- [13] LeCroy, LECROY CORPORATION, WaveSurfer 424, <http://www.lecroy.com>, visited 23/11/2005
- [14] STRUCK INNOVATIVE SYSTEME GMBH. SIS3820 Multi Purpose Scaler, <http://www.struck.de/sis3820.htm>, visited 11/12/2005